[0187] A meta-implementation layer of the present invention may include accessors that access an implementation of an implementation layer or virtual implementations that are the implementation. The accessors may be meta-implementation accessors, language-specific implementation accessors, database specific accessors, and other types of implementation accessors. The meta-implementation layer may be implemented in any language including yet-to-be developed programming languages. The meta-implementation can access any software implementation including yet-to-be developed software implementations. The meta-implementation layer is mapped to a metamodel or other descriptor.

[0188] Rather than requiring an object-oriented class, compiled from a source code written in a specific language, the meta-implementation layer allows a meta-implementation provider to implement the functionality in any way to produce functionality desirable to the system. It makes no difference to the meta-implementation layer how the implementation occurs, as long one meta-implementation accessor exists to access the true implementation for each descriptor in the metamodel.

[0189] In a C++ meta-implementation of the present invention there may be a language-specific implementation model accessor that access a class. The language-specific implementation model accessor may have a one-to-one association relationship with a metamodel, and a zero-to-many association relationships with attribute accessor, operation accessors, signal accessors, constructor accessors and destructor accessors. The class may have a similar one-to-many association relationships with a class attribute, a class method, class event methods, class constructors, and class destructors. The attribute accessors use the class attribute, the operation accessors use the class method, the signal accessors use the class event methods, the constructor accessors use the class construction, and the destructor accessors use the class destructors.

[0190] For example, one meta-implementation provider may write C++ source code for each attribute descriptor, operator, signal, model, and relationship described in the metamodel. Then the meta-implementation provider would generate more C++ source code to create an accessor to each feature on each C++ class.

[0191] For example, a rubber ball class with attributes for bouncy_factor and color and an operation named bounce would be generated as a class with a two member variables, BouncyFactor (one-to-one association relationship, Integer) and Color (one-to-one association relationship, String), and one function, public void bounce( ). Three more classes are created as accessors to these features. RubberBallBouncyFactorAccessor would implement AttributeAccessor and provide access to the bouncy factor attribute of the RubberBall class. RubberBallBouncyColorAccessor would implement AttributeAccessor and provide access to the color attribute of the RubberBall class. RubberBallBounceAction would implement OperationAccessor and execute the bounce function of the RubberBall class. A RubberBallConstructorAccessor would exist to create new instances of RubberBall. The true instances would be wrapped in a RubberBallInstance in order to preserve the connection to the accessor which created it.

[0192] A different meta-implementation provider may generate database tables and store each instance of a metamodel as a record in that table. A database implementation model accessor accesses rows in a database table. Each database implementation model accessor has a zero-to-many association relationship with a metamodel. The database implementation model accessor aggregates attribute accessors, operation accessors, signal accessors, constructor accessors, and destructor accessors which are associated with the metamodel's attribute descriptors, operation descriptors, signal descriptors, constructor descriptors, and destructor descriptors respectively. Each database model accessor instance is a row in a database table. The attribute accessors use the table column, the operation accessors use the stored procedure, the signal accessors use the database triggers or message queues, the constructor accessors use a stored procedure or insert statement, and the destructor accessors use a stored procedure or delete statement.

[0193] To continue the example, the aggregation relationship attributes of a metamodel are stored as columns in the table. Association relationship attributes are stored as foreign key references to records in tables for the associated metamodel. Operations are implemented as stored procedures and interaction relationships would be stored as process entity models in cross-reference tables. A generalized attribute accessor exists which would perform an SQL update statement whenever an attribute accessor requires a change. The attribute accessor performs an SQL select to retrieve the value for the attribute. The meta-implementation vendor may also implement various caching and optimization techniques to reduce unnecessary database access calls.

[0194] A third type of meta-implementation simply acts as a wrapper around a meta-implementation. Due to the limited number of meta-implementation interfaces, it is possible to simply create one wrapper meta-implementation for each interface to add new functionality to all interfaces. Adding new functionality to all aspects of a program is sometimes referred to as aspect programming. New aspects (such as logging all attribute value changes) can be added to a system dynamically at runtime.

[0195] For example, to create a log entry every time an attribute value is changed, a wrapper may be created for the attribute accessor interface that first writes the log entry then calls the normal attribute access method on the true meta-implementation. All implementations of attribute accessor can be wrapped with this attribute accessor wrapper to add this type of functionality.

[0196] Other meta-implementation providers may use LDAP storage, random access files, procedural languages, structured languages, a virtual implementation, or in some yet-to-be developed technology. Meta-implementation providers may use a plain object-oriented implementation, a CORBA implementation, or an Enterprise Java Beans (EJB) implementation approach. The provider may choose to implement the layer using C++ or may opt to use the added facilities made available as part of .NET. A meta-implementation provider may also implement accessors to all of these technologies plus the database and object-oriented technologies described above. By providing accessors to a variety of implementation types, systems will be able to integrate from a variety of sources and distribute the application to take advantage of the particular advantages of each platform.

[0197] A meta-implementation layer of the present invention defines one implementation interface for each descriptor